Please provide complete and well-written solutions to the following exercises.

Due October 18, in the discussion section.

Homework 3

Exercise 1. Let $f \in C^2$. Assume that f has a local maximum at $x \in \mathbf{R}^n$. Conclude that $\nabla f(x) = 0$.

Exercise 2. Prove the following partial converse to a Proposition from the notes.

Let $f \in C^3$. Assume that f has a local minimum at $x \in \mathbf{R}^n$. Then $\nabla f(x) = 0$ and all eigenvalues of $D^2 f(x)$ are nonnegative.

Exercise 3. The Gradient Descent algorithm can behave quite badly for nontrivial reasons. Let $f \colon \mathbf{R} \to \mathbf{R}$ be defined by $f(x) = -x^{1000}$. Suppose we want to maximize f, starting at the initial guess $x^{(0)} = 1$ and using the parameter $\varepsilon = 1/100$. Show that the sequence of points $x^{(0)}, x^{(1)}, \ldots$ does not converge to 0. In fact, show that this sequence of points diverges!

On the other hand, choose a smaller ε to use in the Gradient Descent Algorithm such that the points $x^{(0)}, x^{(1)}, \ldots$ do converge to 0. (For the latter result you can freely use the following fact from analysis: if a sequence of nonnegative numbers $x^{(0)}, x^{(1)}, \ldots$ is strictly decreasing, i.e. if $x^{(i+1)} < x^{(i)}$ for all $i \geq 1$, then the sequence $x^{(0)}, x^{(1)}, \ldots$ converges to a nonnegative real number.)

In this way, a less naive version of the Gradient Descent algorithm will use a small value of ε exactly when the derivative of f is large. That is, the algorithm will adjust ε to depend on $||\nabla f(x^{(n)})||$.

Exercise 4 (Optional)**.** To see an illustration of Newton's Method, see the Applet, Newton Example. In many examples, it only takes a few iterations of the algorithm to get a good approximation for a zero of f.

Write a Matlab program that plots the first few iterates of Newton's method, as in the above applet. (Or, use a different programming language if you prefer it.) You can test your program using, e.g. the function $f(x) = x^2 - 1$ with initial guess $x_0 = 2$.

Exercise 5. Recall that Newton's method is an algorithm for finding zeros of a function f. It consists in iterating the map

$$\varphi(x) := x - (f'(x))^{-1} f(x) .$$

Thus, we start with some given $x^{(0)}$ and define $x^{(1)} := \varphi(x^{(0)})$, $x^{(2)} := \varphi(x^{(1)})$, etc.

This problem is devoted to an analysis of the convergence of Newton's method. For simplicity, we work in one dimension, i.e. we set $n = 1$. Without loss of generality, we assume that the zero of f we are interested in is at the origin: $f(0) = 0$. We shall show that, assuming

$f'(0)$ is invertible and $f \in C^2$, the sequence $x^{(0)}, x^{(1)}, \ldots$ converges to 0 provided $x^{(0)}$ is close enough to 0.

Let $K > 1$, $R > 0$ and suppose that

$$K^{-1} \leq |f'(x)| \leq K, \qquad |f''(x)| \leq K, \qquad \forall\, |x| \leq R. \tag{1}$$

(i) Using Taylor's Theorem, show that there exists $\varepsilon > 0$ such that, if $|x| < \varepsilon$, then
$$|f(0) - (f(x) - xf'(x))| \leq K\,|x|^2\,.$$

(ii) Suppose $|x^{(0)}| < \varepsilon$. Then from part (i), $|f(x^{(0)}) - x^{(0)}f'(x^{(0)}))| \leq K|x^{(0)}|^2$. Using the definition of $x^{(1)}$, deduce that
$$|x^{(1)}| \leq K|f(x^{(0)}) - x^{(0)}f'(x^{(0)}))| \leq K^2|x^{(0)}|^2.$$

(iii) Let $n \geq 1$. Suppose $|x^{(n)}| < \varepsilon$. Show that
$$|x^{(n+1)}| \leq K^2|x^{(n)}|^2.$$

(iv) Conclude that $\lim_{n\to\infty} |x^{(n)}| = 0$, as desired.

Exercise 6 (Optional)**.** Write a Matlab program to implement Newton's Method for functions of two variables. Test your implementation on Rosenbrock's function: $f(x, y) = 100(y - x^2)^2 + (1 - x)^2$.

Exercise 7. Let A be an $n \times n$ symmetric positive definite matrix. Let $y, z \in \mathbf{R}^n$. Define a function $\langle\cdot,\cdot\rangle_A \colon \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}$ by
$$\langle y, z\rangle_A := y^T Az.$$
Show that $\langle\cdot,\cdot\rangle_A$ is an inner product on $\mathbf{R}^n$. That is, show:

- For any $y \in \mathbf{R}^n$ with $y \neq 0$, $\langle y, y\rangle_A > 0$.
- For any $y, z \in \mathbf{R}^n$, $\langle y, z\rangle_A = \langle z, y\rangle_A$.
- For any $y, z \in \mathbf{R}^n, \lambda \in \mathbf{R}$, $\langle \lambda y, z\rangle_A = \lambda\langle y, z\rangle_A$.
- For any $w, y, z \in \mathbf{R}^n, \lambda \in \mathbf{R}$, $\langle w + y, z\rangle_A = \langle w, z\rangle_A + \langle y, z\rangle_A$.

Then, a standard fact from linear algebra implies that the following function is a norm on $\mathbf{R}^n$.
$$||y||_A := \sqrt{\langle y, y\rangle_A} = \sqrt{y^T Ay}.$$

Exercise 8. At iteration i, the Conjugate Gradient (CG) Algorithm from the notes only requires the values of $x^{(i)}, x^{(i-1)}$, $r^{(i)}, r^{(i-1)}$ and $d^{(i)}, d^{(i-1)}$. So, the memory storage requirement of this algorithm is fairly small

Give a bound on the minimum amount of numbers that the CG Algorithm needs to store in memory, over the entire duration of the algorithm.

Compared to the Basic CG Algorithm, the CG Algorithm requires a few less matrix multiplications at each iteration, which saves some time.

Give a bound on the total number of arithmetic operations that the CG Algorithm performs over the duration of the entire algorithm. Your bound should depend on the size n of the matrix A and on the number m of nonzero entries of A. (Your bound should be something like mn)

Exercise 9 (Optional)**.** Implement the Conjugate Gradient Algorithm in Matlab. Compare your algorithm to Matlab's `pcg` function.